CHAPTER
18

# Configuring an FTP Server

## IN THIS CHAPTER

Learning how FTP works

Getting a vsftpd server installed

Choosing security settings for vsftpd

Setting up vsftpd configuration files

Running FTP clients

The *File Transfer Protocol (FTP)* is one of the oldest protocols in existence for sharing files over networks. Although there are more secure protocols for network file sharing, FTP is still used quite often for making files freely available on the Internet.

Several FTP server projects are available with Linux today. However, the one often used with Fedora, Red Hat Enterprise Linux, CentOS, Ubuntu, and other Linux distributions is the Very Secure FTP Daemon (`vsftpd` package). This chapter describes how to install, configure, use, and secure an FTP server using the `vsftpd` package.

## Understanding FTP

FTP operates in a client/server model. An FTP server daemon listens for incoming requests (on TCP port 21) from FTP clients. The client presents a login and password. If the server accepts the login information, the client can interactively traverse the filesystem, list files and directories, and then download (and sometimes upload) files.

What makes FTP insecure is that everything sent between the FTP client and server is done in clear text. The FTP protocol was created at a time when most computer communication was done on private lines or over dial-up, where encryption was not thought to be critical. If you use FTP over a public network, someone sniffing the line anywhere between the client and server would be able to see not only the data being transferred but also the authentication process (login and password information).

So, FTP is not good for sharing files privately (use SSH commands such as `sftp`, `scp`, or `rsync` if you need private, encrypted file transfers). However, if you are sharing public documents, open source software repositories, or other openly available data, FTP is a good choice. Regardless of the operating system people use, they surely have an FTP file transfer application available to get files that you offer from your FTP server.

When users authenticate to an FTP server in Linux, their usernames and passwords are authenticated against the standard Linux user accounts and passwords. There is also a special, non-authenticated account used by the FTP server called anonymous. The anonymous account can be accessed by anyone because it does not require a valid password. In fact, the term *anonymous FTP server* is often used to describe a public FTP server that does not require (or even allow) authentication of a legitimate user account.

**Note**

Although the ability to log in to the `vsftpd` server using a regular Linux user account is enabled by default in Fedora and Red Hat Enterprise Linux, if SELinux is set to enforcing mode, it prevents the logins and file transfers from succeeding. If you want to keep SELinux in enforcing mode yet still allow Linux logins, you can change a Boolean (see the section "Configuring SELinux for your FTP server" later in this chapter) to allow regular user logins to succeed.

After the authentication phase (on the control port, TCP port 21), a second connection is made between the client and server. FTP supports both active and passive connection types. With an *active FTP connection*, the server sends data from its TCP port 20 to some random port the server chooses above port 1023 on the client. With a *passive FTP connection*, the client requests the passive connection and requests a random port from the server.

Many browsers support passive FTP mode so that if the client has a firewall, it doesn't block the data port that the FTP server might use in active mode. Supporting passive mode requires some extra work on the server's firewall to allow random connections to ports above 1023 on the server. The section "Opening up your firewall for FTP," later in this chapter, describes what you need to do to your Linux firewall to make both passive and active FTP connections work.

After the connection is established between the client and server, the client's current directory is established. For the anonymous user, the `/var/ftp` directory is the home directory for Fedora or RHEL, and it's `/srv/ftp` for Ubuntu and most Debian-based distributions. The anonymous user cannot go outside of the `/var/ftp` directory structure. If a regular user, let's say joe, logs in to the FTP server, `/home/joe` is joe's current directory, but joe can change to any part of the filesystem for which he has permission.

Command-oriented FTP clients (such as `lftp` and `ftp` commands) go into an interactive mode after connecting to the server. From the prompt you see, you can run many commands that are similar to those that you would use from the shell. You could use `pwd` to see your current directory, `ls` to list directory contents, and `cd` to change directories. When you see a file that you want, you use the `get` and `put` commands to download files from or upload them to the server, respectively.

With graphical tools for accessing FTP servers (such as a web browser), you type the URL of the site that you want to visit (such as `ftp://docs.example.com`) into the location box of the browser. If you don't add a username or password, an anonymous connection is made and the contents of the home directory of the site are displayed. Click links to directories to change to those directories. Click links to files to display or download those files to your local system.

Armed with some understanding of how FTP works, you are now ready to install an FTP server (vsftpd package) on your Linux system.

## Installing the vsftpd FTP Server

Setting up the Very Secure FTP server requires only one package in Fedora, RHEL, and other Linux distributions: vsftpd. Assuming you have a connection to your software repository, just type the following as root for Fedora or RHEL to install vsftpd:

```
# yum install vsftpd
```

If you are using Ubuntu (or another Linux distribution based on Debian packaging), type the following to install vsftpd:

```
$ sudo apt-get install vsftpd
```

Here are some commands that you can run after the vsftpd package is installed to familiarize yourself with the contents of that package. From Fedora or RHEL, run this command to get some general information about the package:

```
# rpm -qi vsftpd
...
Packager    : Fedora Project
Vendor      : Fedora Project
URL         : https://security.appspot.com/vsftpd.html
Summary     : Very Secure Ftp Daemon
Description : vsftpd is a Very Secure FTP daemon. It was written
              completely from scratch.
```

If you want to get more information about vsftpd, follow the URL listed to the related website (https://security.appspot.com/vsftpd.html). You can get additional documentation and information about the latest revisions of vsftpd.

You can view the full contents of the vsftpd package (rpm -ql vsftpd), or you can view just the documentation (-qd) or configuration (-qc) files. To see the documentation files in the vsftpd package, use the following:

```
# rpm -qd vsftpd
/usr/share/doc/vsftpd/EXAMPLE/INTERNET_SITE/README
...
/usr/share/doc/vsftpd/EXAMPLE/PER_IP_CONFIG/README
...
/usr/share/doc/vsftpd/EXAMPLE/VIRTUAL_HOSTS/README
/usr/share/doc/vsftpd/EXAMPLE/VIRTUAL_USERS/README
...
/usr/share/doc/vsftpd/FAQ
...
/usr/share/doc/vsftpd/vsftpd.xinetd
/usr/share/man/man5/vsftpd.conf.5.gz
/usr/share/man/man8/vsftpd.8.gz
```

In the /usr/share/doc/vsftpd/EXAMPLE directory structure, there are sample configuration files included to help you configure vsftpd in ways that are appropriate for an Internet site, multiple IP address site, and virtual hosts. The main /usr/share/doc/vsftpd directory contains an FAQ (frequently asked questions), installation tips, and version information.

The man pages might have the most useful information when you set out to configure the vsftpd server. Type **man vsftpd.conf** to read about the configuration file and **man vsftpd** to read about the daemon process and how to manage it as a systemd service.

To list the configuration files, type the following:

```
# rpm -qc vsftpd
/etc/logrotate.d/vsftpd
/etc/pam.d/vsftpd
/etc/vsftpd/ftpusers
/etc/vsftpd/user_list
/etc/vsftpd/vsftpd.conf
```

The main configuration file is /etc/vsftpd/vsftpd.conf (in RHEL and Fedora) or /etc/vsftpd.conf (in Ubuntu). The ftpusers and user _ list (Fedora and RHEL, but not Ubuntu) files in the same directory store information about user accounts that are restricted from accessing the server. The /etc/pam.d/vsftpd file sets how authentication is done to the FTP server. The /etc/logrotate.d/vsftpd file configures how log files are rotated over time.

Now you have vsftpd installed and have taken a quick look at its contents. The next step is to start up and test the vsftpd service.

## Starting the vsftpd Service

No configuration is required to launch the vsftpd service if you just want to use the default settings. If you start vsftpd as it is delivered with Fedora, the following is what you get:

- The vsftpd service starts the vsftpd daemon, which runs in the background.
- The standard port on which the vsftpd daemon listens is TCP port 21. By default, data is transferred to the user, after the connection is made, on TCP port 20. TCP port 21 must be open in the firewall to allow new connections to access the service. Both IPv4 and IPv6 connections are available by default. This procedure changes to the TCP IPv4 service. (See the section "Securing Your FTP Server" later in this chapter for details on opening ports, enabling connection tracking needed for passive FTP, and setting other firewall rules related to FTP.)
- The vsftpd daemon reads vsftpd.conf to determine what features the service allows.
- Linux user accounts (excluding administrative users) can access the FTP server. The anonymous user account (no password required) can be enabled. (If SELinux is in enforcing mode, you need to set a Boolean to allow regular users to log in to the FTP server. See the section "Securing Your FTP Server" for details.)

- The anonymous user has access only to the /var/ftp directory and its subdirectories. A regular user starts with their home directory as the current directory but can access any directory to which the user would be able to gain access via a regular login or SSH session. Lists of users in the /etc/vsftpd/user _ list and /etc/vsftpd/ftpusers files define some administrative and special users who do not have access to the FTP server (root, bin, daemon, and others).
- By default, the anonymous user can download files from the server but not upload them. A regular user can upload or download files, based on regular Linux permissions.
- Log messages detailing file uploads or downloads are written in the /var/log/xferlogs file. Those log messages are stored in a standard xferlog format.

If you are ready to start your server using the defaults just described, the following examples show you how to do that. If you want to change some additional settings first, go to the section "Configuring Your FTP Server," later in this chapter, finalize your settings, and then come back here for instructions on how to enable and start your server.

1. Check the vsftpd service. Before you start the vsftpd service, you can check out whether it is running already. In Fedora or Red Hat Enterprise Linux 7 or 8, you do the following:

```
        # systemctl status vsftpd.service
        vsftpd.service - Vsftpd ftp daemon
               Loaded: loaded (/usr/lib/systemd/system/vsftpd.
service; disabled)
               Active: inactive (dead)
```

   In Red Hat Enterprise Linux 6, you need two commands to see the same information:

```
        # service vsftpd status
        vsftpd is stopped
        # chkconfig --list vsftpd
        vsftpd  0:off  1:off  2:off  3:off  4:off  5:off  6:off
```

   In both the Fedora and RHEL examples above, the service, chkconfig, and systemctl commands show the status as stopped. You can also see that it is disabled in Fedora and RHEL 7 or 8 and off at every runlevel for RHEL 6. Disabled (off) means that the service will not turn on automatically when your start the system.

2. To start and enable vsftpd in Fedora or RHEL 7 or 8 (then check the status), type the following:

```
        # systemctl start vsftpd.service
        # systemctl enable vsftpd.service
        ln -s '/lib/systemd/system/vsftpd.service'
          '/etc/systemd/system/multi-user.target.wants/vsftpd.
service'
        # systemctl status vsftpd.service
        vsftpd.service - Vsftpd ftp daemon
               Loaded: loaded (/usr/lib/systemd/system/vsftpd.
service;
                  enabled vendor preset: disabled))
               Active: active (running) since Wed, 2019-09-18
00:09:54 EDT; 22s ago
               Main PID: 4229 (vsftpd)
                  Tasks: 1 (limit: 12232)
                 Memory: 536.0K
                  CGroup: /system.slice/vsftpd.service
               └4229 /usr/sbin/vsftpd /etc/vsftpd/vsftpd.conf
```

In Red Hat Enterprise Linux 6, start and turn on (enable) `vsftpd` (then check the status), as follows:

```
        # service vsftpd start
        Starting vsftpd for vsftpd:                  [  OK  ]
        # chkconfig vsftpd on ; chkconfig --list vsftpd
        vsftpd       0:off    1:off    2:on    3:on    4:on    5:on
6:off
```

3. Now, on either system, you could check that the service is running using the `netstat` command:

```
        # netstat -tupln | grep vsftpd
        tcp  0   0 0.0.0.0:21  0.0.0.0:*  LISTEN   4229/vsftpd
```

   From the `netstat` output, you can see that the `vsftpd` process (process ID of 4229) is listening (`LISTEN`) on all IP addresses for incoming connections on port 21 (0.0.0.0:21) for the TCP (`tcp`) protocol.

4. A quick way to check that `vsftpd` is working is to put a file in the `/var/ftp` directory and try to open it from your web browser on the local host:

```
        # echo "Hello From Your FTP Server" > /var/ftp/hello.txt
```

   From a web browser on the local system, type the following into the location box of Firefox or another browser:

```
        ftp://localhost/hello.txt
```

If the text `Hello From Your FTP Server` appears in the web browser, the `vsftpd` server is working and accessible from your local system. Next, try this again from a web browser on another system, replacing `localhost` with your host's IP address or fully qualified host name. If that works, the `vsftpd` server is publicly accessible. If it doesn't, which it quite possibly may not, see the next section, "Securing Your FTP Server." That section tells you how to open firewalls and modify other security features to allow access and otherwise secure your FTP server.

## Securing Your FTP Server

Even though it is easy to get a `vsftpd` FTP server started, that doesn't mean that it is immediately fully accessible. If you have a firewall in place on your Linux system, it is probably blocking access to all services on your system except for those that you have explicitly allowed.

If you decide that the default `vsftpd` configuration works for you as described in the previous section, you can set to work allowing the appropriate access and providing security for your `vsftpd` service. To help you secure your `vsftpd` server, the next sections describe how to configure your firewall and SELinux (Booleans and file contexts).

### Opening up your firewall for FTP

If you have a firewall implemented on your system, you need to add firewall rules that allow incoming requests to your FTP site and allow packets to return to your system on established connections. Firewalls are implemented using `iptables` rules and managed with the `iptables` service or `firewalld` service (see Chapter 25, "Securing Linux on a Network," for details about firewall services).

In Fedora and Red Hat Enterprise Linux, firewall rules have traditionally been stored in the `/etc/sysconfig/iptables` file and the underlying service was `iptables` (RHEL 6) or `iptables.service` (Fedora). Modules are loaded into your firewall from the `/etc/sysconfig/iptables-config` file. In RHEL 7 and Fedora 21 or later, the new `firewalld` service manages those rules and rules are stored in the `/etc/firewalld/zones` directory.

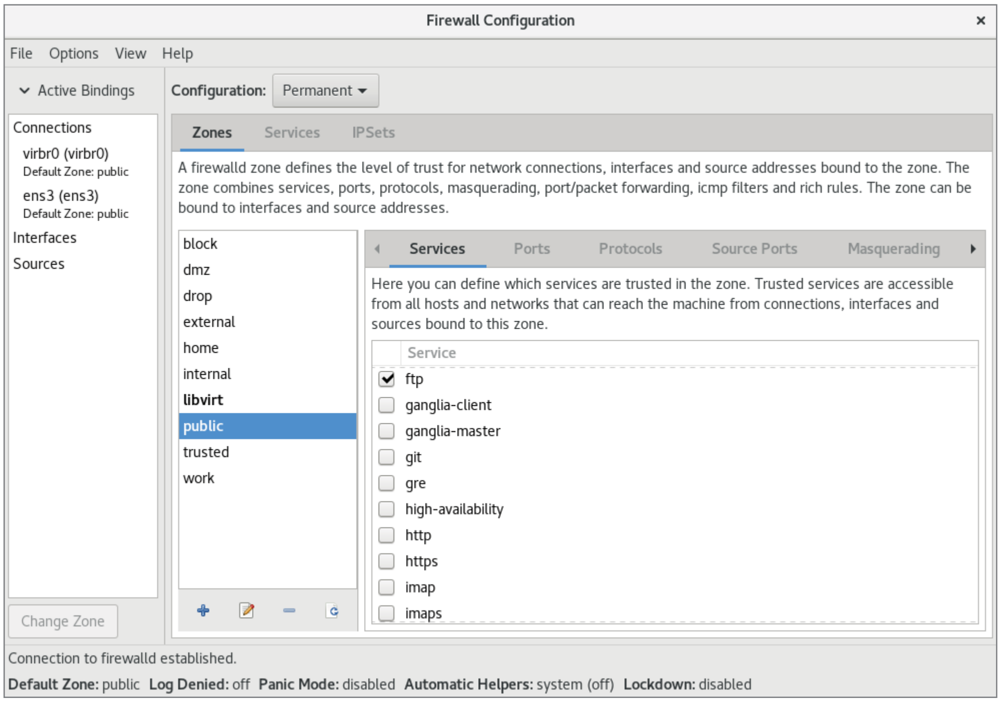
**NOTE**

**It is best to work on your firewall directly from a system console, if possible, instead of over a remote login (such as `ssh`) because a small error can immediately lock you out of your server. After that, you must go over to the console to get back into the server and fix the problem. You need to add a few things to your firewall to allow access to your FTP server without opening up access to other services. First, you need to allow your system to accept requests on TCP port 21; then you need to make sure that the connection tracking module is loaded.**

In RHEL 7 and Fedora 20 or later, you can use the Firewall Configuration window to enable your firewall and open access to your FTP service. Install the `firewall-config` package and run `firewall-config` to start the Firewall Configuration window, as shown in Figure 18.1.

**FIGURE 18.1**

Open access to your FTP service from the Firewall Configuration window.



A firewalld zone defines the level of trust for network connections, interfaces and source addresses bound to the zone. The zone combines services, ports, protocols, masquerading, port/packet forwarding, icmp filters and rich rules. The zone can be bound to interfaces and source addresses.

Here you can define which services are trusted in the zone. Trusted services are accessible from all hosts and networks that can reach the machine from connections, interfaces and sources bound to this zone.

Connection to firewalld established.

**Default Zone:** public **Log Denied:** off **Panic Mode:** disabled **Automatic Helpers:** system (off) **Lockdown:** disabled

Next, to open access permanently to your FTP service, click the Configuration box and select Permanent. Then select the check box next to ftp under the Services tab. This automatically opens TCP port 21 (FTP) on your firewall and loads kernel modules needed to allow access to passive FTP service. Select Options ➪ Reload Firewalld to apply the firewall rule permanently.

For RHEL 6 and earlier systems, add rules directly to the `/etc/sysconfig/iptables` file. If you are using a default firewall, rules in the beginning open access to requests for any services coming from the local host and allow packets to come in that are associated with, or related to, established connections. In the middle are rules that open ports for service requests that you have already allowed, such as the secure shell service (`sshd` on TCP port 22). At the end of the rules, a final rule usually `DROP`s or `REJECT`s any request that has not explicitly been allowed.

To allow public access to someone requesting your FTP server, you want to allow new requests to TCP port 21. You typically want to add the rule somewhere before the final `DROP` or `REJECT` rule. The following output shows partial contents of the `/etc/sysconfig/iptables` file with the rule allowing access to your FTP server in bold:

```
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 21 -j ACCEPT
...
-A INPUT -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

This example shows that, for the filter table, the firewall accepts packets from established connections, connections from local hosts, and any new requests on TCP port 22 (SSH service). The line we just added (`--dport 21`) allows any packets on new connections to TCP port 21 to be accepted.

**NOTE**
It is important to have the `ESTABLISHED, RELATED` line in your `iptables` firewall rules. Without that line, users would be able to connect to your SSH (port 22) and FTP (port 21) services, but they would not be able to communicate after that. So, a user could get authenticated but not be able to transfer data.

The next thing that you must do on RHEL 6 and earlier systems is set up the FTP connection tracking module to be loaded each time the firewall starts up. Edit this line at the beginning of the `/etc/sysconfig/iptables-config` file to appear as follows:

```
IPTABLES_MODULES="nf_conntrack_ftp"
```

At this point, you can restart your firewall (keeping in mind that a mistake could lock you out if you are logged in remotely). Use one of the following two commands to restart your firewall, depending on whether your system is using the older `iptables` service or the newer `firewalld` service:

```
# service iptables restart
```

*or*

```
# systemctl restart firewalld.service
```

Try again to access your FTP server from a remote system (using a web browser or some other FTP client).

## Configuring SELinux for your FTP server

If SELinux is set to permissive or disabled, it does not block access to the `vsftpd` service in any way. However, if SELinux is in enforcing mode, a few SELinux issues could cause your `vsftpd` server not to behave as you would like. Use the following commands to check the state of SELinux on your system:

```
# getenforce
Enforcing
# grep ^SELINUX= /etc/sysconfig/selinux
SELINUX=enforcing
```

The `getenforce` command shows how SELinux is currently set. (Here, it's in enforcing mode.) The `SELINUX=` variable in `/etc/sysconfig/selinux` shows how SELinux is set when the system comes up. If it is in enforcing mode, as it is here, check the `ftpd _ selinux` man page for information about SELinux settings that can impact the operation of your `vsftpd` service. Install the `selinux-policy-doc` package to get the `ftpd _ selinux` man page as well as man pages for other services with SELinux policies.

Here are some examples of file contexts that must be set for SELinux to allow files and directories to be accessed by `vsftpd`:

- To share content so that it can be downloaded to FTP clients, that content must be marked with a `public_content_t` file context. Files created in the `/var/ftp` directory or its subdirectories inherit `public_content_t` file context automatically. (Be sure to create new content or copy existing content to the `/var/ftp` directories. Moving the files there may not change the file context properly.)
- To allow files to be uploaded by anonymous users, the file context on the directory to which you upload must be set to `public_content_rw_t`. (Other permissions, SELinux Booleans, and `vsftpd.conf` settings must be in place for this to work as well.)

If you have files in the `/var/ftp` directory structure that have the wrong file contexts (which can happen if you move files there from other directories instead of copying them), you can change or restore the file context on those files so that they can be shared. For example, to recursively change the file context of the `/var/ftp/pub/stuff` directory so that the content can be readable from the FTP server through SELinux, enter the following:

```
# semanage fcontext -a -t public_content_t "/var/ftp/pub/stuff(/.*)?"
# restorecon -F -R -v /var/ftp/pub/stuff
```

If you wanted to allow users also to write to a directory as well as to read from it, you would need to assign the `public _ content _ rw _ t` file context to the directory to which you want to allow uploads. This example tells SELinux to allow uploading of files to the `/var/ftp/pub/uploads` directory:

```
# semanage fcontext -a -t public_content_rw_t\
   "/var/ftp/pub/uploads(/.*)?"
# restorecon -F -R -v /var/ftp/pub/uploads
```

FTP server features that are considered insecure by SELinux have Booleans that let you allow or disallow those features. Here are some examples:

- For SELinux to allow anonymous users to read and write files and directories, you need to turn on the `allow _ ftpd _ anon _ write` (RHEL 6) or `ftpd _ anon _ write` (RHEL 7 or later) Boolean:

  ```
  # setsebool -P ftpd_anon_write on
  ```

- To be able to mount remote NFS or CIFS (Windows) shared filesystems and share them from your vsftpd server, you need to turn on the following two Booleans, respectively:

```
# setsebool -P allow_ftpd_use_nfs on
# setsebool -P allow_ftpd_use_cifs on
```

If you ever find that you cannot access files or directories from your FTP server that you believe should be accessible, try turning off SELinux temporarily:

```
# setpenforce 0
```

If you can access the files or directories with SELinux now in permissive mode, put the system back in enforcing mode (setenforce 1). Now you know you have to go back through your SELinux settings and find out what is preventing access. (See Chapter 24, "Enhancing Linux Security with SELinux," for more information on SELinux.)

### Relating Linux file permissions to vsftpd

The vsftpd server relies on standard Linux file permissions to allow or deny access to files and directories. As you would expect, for an anonymous user to view or download a file, at least read permission must be open for other (------r--). To access a directory, at least execute permission must be on for other (--------x).

For regular user accounts, the general rule is that if a user can access a file from the shell, that user can access the same file from an FTP server. So, typically, regular users should at least be able to get (download) and put (upload) files to and from their own home directories, respectively. After permissions and other security provisions are in place for your FTP server, you may want to consider other configuration settings for your FTP server.

## Configuring Your FTP Server

Most of the configuration for the vsftpd service is done in the /etc/vsftpd/vsftpd .conf file. Examples of vsftpd.conf for different types of sites are included in the /usr/share/doc/vsftpd directory. Depending on how you want to use your FTP site, the following sections discuss a few ways to configure your FTP server.

Remember to restart the vsftpd service after making any configuration changes.

### Setting up user access

The vsftpd server comes with all local Linux users (those listed in the /etc/passwd file) configured to access the server and the anonymous user prevented. This is based on the following vsftpd.conf settings:

```
anonymous_enable=NO
local_enable=YES
```

Some web server companies let users use FTP to upload the content for their own web servers. In some cases, the users have FTP-only accounts, meaning that they cannot log in to a shell, but they can log in via FTP to manage their content. Creating a user account that has no default shell (actually, `/sbin/nologin`) is how you can keep a user from logging into a shell but still allow FTP access. For example, the `/etc/passwd` entry for the FTP-only user account `bill` might look something like the following:

```
bill:x:1000:1000:Bill Jones:/home/bill:/sbin/nologin
```

With the user account set with `/sbin/nologin` as the default shell, any attempts to log in from a console or via `ssh` as the user `bill` are denied. However, as long as `bill` has a password and local account access to the FTP server is enabled, `bill` should be able to log in to the FTP server via an FTP client.

Not every user with an account on the Linux system has access to the FTP server. The setting `userlist _ enable=YES` in `vsftpd.conf` says to deny access to the FTP server to all accounts listed in the `/etc/vsftpd/user _ list` file. That list includes administrative users `root`, `bin`, `daemon`, `adm`, `lp`, and others. You can add to that list other users to whom you would like to deny access.

If you change `userlist _ enable` to `NO`, the `user _ list` file becomes a list of only those users who do have access to the server. In other words, setting `userlist _ enable=NO`, removing all usernames from the `user _ list` file, and adding the usernames `chris`, `joe`, and `mary` to that file cause the server to allow only those three users to log in to the server.

No matter how the value of `userlist _ enable` is set, the `/etc/vsftpd/ftpusers` file always includes users who are denied access to the server. Like the `userlist _ enable` file, the `ftpusers` file includes a list of administrative users. You can add more users to that file if you want them to be denied FTP access.

One way to limit access to users with regular user accounts on your system is to use `chroot` settings. Here are examples of some `chroot` settings:

```
chroot_local_user=YES
chroot_list_enable=YES
chroot_list_file=/etc/vsftpd/chroot_list
```

With the settings just shown uncommented, you could create a list of local users and add them to the `/etc/vsftpd/chroot _ list` file. After one of those users logged in, that user would be prevented from going to places in the system that were outside of that user's home directory structure.

If uploads to your FTP server are allowed, the directories a user tries to upload to must be writeable by that user. However, uploads can be stored under a username other than that of the user who uploaded the file. This is one of the features discussed next, in the section "Allowing uploading."

## Allowing uploading

To allow any form of writing to the `vsftpd` server, you must have `write _ enable=YES` set in the `vsftpd.conf` file (which it is, by default). Because of that, if local accounts are enabled, users can log in and immediately begin uploading files to their own home directories. However, anonymous users are denied the ability to upload files by default.

To allow anonymous uploads with `vsftpd`, you must have the first option in the following code example, and you may want the second line of code as well (both can be enabled by uncommenting them from the `vsftpd.conf` file). The first allows anonymous users to upload files; the second allows them to create directories:

```
anon_upload_enable=YES
anon_mkdir_write_enable=YES
```

The next step is to create a directory where anonymous users can write. Any directory under the `/var/ftp` directory that has write permissions for the user `ftp`, the `ftp` group, or `other` can be written to by an anonymous user. A common thing is to create an uploads directory with permission open for writing. The following are examples of commands to run on the server:

```
# mkdir /var/ftp/uploads
# chown ftp:ftp /var/ftp/uploads
# chmod 775 /var/ftp/uploads
```

As long as the firewall is open and SELinux Booleans are set properly, an anonymous user can `cd` to the uploads directory and put a file from the user's local system into the uploads directory. On the server, the file would be owned by the `ftp` user and `ftp` group. The permissions set on the directory (775) would allow you to see the files that were uploaded but not change or overwrite them.

One reason for allowing anonymous FTP, and then enabling it for anonymous uploads, is to allow people you don't know to drop files into your uploads folder. Because anyone who can find the server can write to this directory, some form of security needs to be in place. You want to prevent an anonymous user from seeing files uploaded by other users, taking files, or deleting files uploaded by other anonymous FTP users. One form of security is the `chown` feature of FTP.

By setting the following two values, you can allow anonymous uploads. The result of these settings is that when an anonymous user uploads a file, that file is immediately assigned ownership of a different user. The following is an example of some `chown` settings that you could put in your `vsftpd.conf` file to use with your anonymous upload directory:

```
chown_uploads=YES
chown_username=joe
```

If an anonymous user were to upload a file after `vsftpd` was restarted with these settings, the uploaded file would be owned by the user `joe` and the `ftp` group. Permissions would be read/write for the owner and nothing for anyone else (`rw-------`).

So far, you have seen configuration options for individual features on your vsftpd server. Some sets of vsftp.conf variables can work together in ways that are appropriate for certain kinds of FTP sites. The next section contains one of these examples, represented by a sample vsftpd.conf configuration file that comes with the vsftpd package. That file can be copied from a directory of sample files to the /etc/vsftpd/vsftpd.conf file to use for an FTP server that is available on the Internet.

## Setting up vsftpd for the Internet

To share files from your FTP server safely to the Internet, you can lock down your server by limiting it to only allow downloads and only from anonymous users. To start with a configuration that is designed to share vsftpd files safely over the Internet, back up your current /etc/vsftpd/vsftpd.conf file and copy this file to overwrite your vsftpd.conf:

```
/usr/share/doc/vsftpd/EXAMPLE/INTERNET_SITE/vsftpd.conf
```

The following paragraphs describe the contents of that vsftpd.conf. Settings in the first section set the access rights for the server:

```
# Access rights
anonymous_enable=YES
local_enable=NO
write_enable=NO
anon_upload_enable=NO
anon_mkdir_write_enable=NO
anon_other_write_enable=NO
```

Turning on anonymous _ enable (YES)and turning off local _ enable (NO)ensures that no one can log in to the FTP server using a regular Linux user account. Everyone must come in through the anonymous account. No one can upload files (write _ enable=NO). Then, the anonymous user cannot upload files (anon _ upload _ enable=NO), create directories (anon _ mkdir _ write _ enable=NO), or otherwise write to the server (anon _ other _ write _ enable=NO). Here are the Security settings:

```
# Security
anon_world_readable_only=YES
connect_from_port_20=YES
hide_ids=YES
pasv_min_port=50000
pasv_max_port=60000
```

Because the vsftpd daemon can read files assigned to the ftp user and group, setting anon _ world _ readable _ only=YES ensures that anonymous users can see files where the read permission bit is turned on for other (------r--), but not write files. The connect _ from _ port _ 20=YES setting gives the vsftpd daemon slightly more permission to send data the way a client might request by allowing PORT-style data communications.

Using hide _ ids=YES hides the real permissions set on files, so to the user accessing the FTP site, everything appears to be owned by the ftp user. The two pasv settings restrict the range of ports that can be used with passive FTP (where the server picks a higher number port on which to send data) to between 50000 and 60000.

The next section contains features of the vsftpd server:

```
# Features
xferlog_enable=YES
ls_recurse_enable=NO
ascii_download_enable=NO
async_abor_enable=YES
```

With xferlog _ enable=YES, all file transfers to and from the server are logged to the /var/log/xferlog file. Setting ls _ recurse _ enable=NO prevents users from recursively listing the contents of an FTP directory (in other words, it prevents the type of listing that you could get with the ls -R command) because on a large site, that could drain resources. Disabling ASCII downloads forces all downloads to be in binary mode (preventing files from being translated in ASCII, which is inappropriate for binary files). The async _ abor _ enable=YES setting ensures that some FTP clients, which might hang when aborting a transfer, will not hang.

The following settings have an impact on performance:

```
# Performance
one_process_model=YES
idle_session_timeout=120
data_connection_timeout=300
accept_timeout=60
connect_timeout=60
anon_max_rate=50000
```

With one _ process _ model=YES set, performance can improve because vsftpd launches one process per connection. Reducing the idle _ session _ timeout from the default 300 seconds to 120 seconds causes FTP clients that are idle more than 2 minutes to be disconnected. Thus, less time is spent managing FTP sessions that are no longer in use. If a data transfer stalls for more than data _ connection _ timeout seconds (300 seconds here), the connection to the client is dropped.

The accept _ timeout setting of 60 seconds allows 1 minute for a PASV connection to be accepted by the remote client. The connect _ timeout sets how long a remote client has to respond to a request to establish a PORT-style data connection. Limiting the transfer rate to 50000 (bytes per second) with anon _ max _ rate can improve overall performance of the server by limiting how much bandwidth each client can consume.

## Using FTP Clients to Connect to Your Server

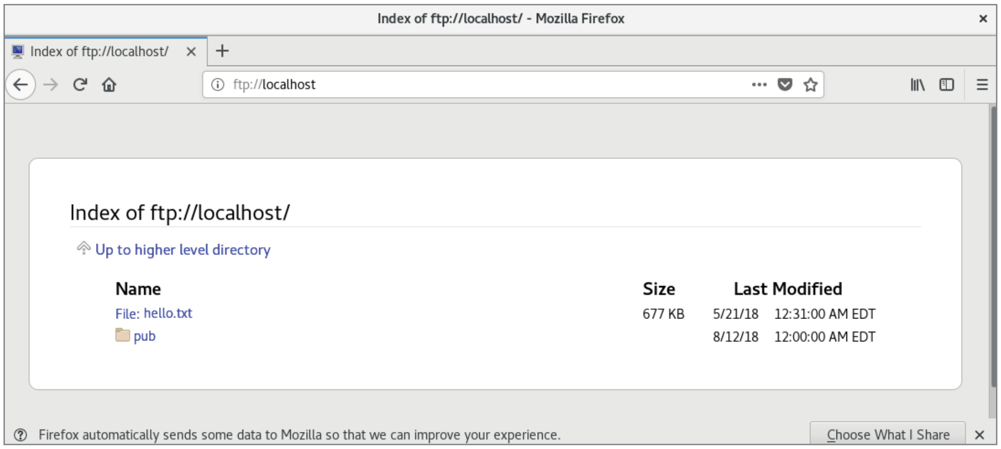
Many client programs come with Linux, which you can use to connect to your FTP server. If you simply want to do an anonymous download of some files from an FTP server, your Firefox web browser provides an easy interface to do that. For more complex interactions between your FTP client and server, you can use command-line FTP clients. The following sections describe some of these tools.

## Accessing an FTP server from Firefox

The Firefox web browser provides a quick and easy way to test access to your FTP server or to access any public FTP server. On your own system, type **ftp://localhost** into the location box. You are prompted to log in, which you can do as a regular user or the anonymous user if your server is accessible via anonymous FTP. As the anonymous user, you should see something similar to the example shown in Figure 18.2.

**FIGURE 18.2**

Accessing an FTP server from Firefox


Index of ftp://localhost/ - Mozilla Firefox
Index of ftp://localhost/
ftp://localhost
Choose What I Share

Index of ftp://localhost/

Up to higher level directory

| Name | Size | Last Modified |
|---|---|---|
| File: hello.txt | 677 KB | 5/21/18 12:31:00 AM EDT |
| pub | | 8/12/18 12:00:00 AM EDT |

Firefox automatically sends some data to Mozilla so that we can improve your experience.

To log in to an FTP server as a particular user from Firefox, you can precede the host name with a `username:password@` notation, as shown in the following example:

```
ftp://chris:MypassWd5@localhost
```

If you provide the correct username and password, you should immediately see the contents of your home directory. Click a folder to open it. Click a file to download or view the file.

## Accessing an FTP server with the lftp command

To test your FTP server from the command line, you can use the `lftp` command. To install the `lftp` command in Fedora or RHEL, enter the following from the command line:

```
# yum install lftp
```

If you use the `lftp` command with just the name of the FTP server you are trying to access, the command tries to connect to the FTP server as the anonymous user. By adding the `-u` *username*, you can enter the user's password when prompted and gain access to the FTP server as the user you logged in as.

After you have added your user and password information, you get an lftp prompt, ready for you to start typing commands. The connection is made to the server when you type your first command. You can use the commands to move around the FTP server and then use the get and put commands to download and upload files.

The following example shows how to use commands as just described. It assumes that the FTP server (and associated security measures) has been configured to allow local users to connect and to read and write files:

```
# lftp -u chris localhost
Password:
********
lftp chris@localhost:~> pwd
ftp://chris@localhost/%2Fhome/chris
lftp chris@localhost:~> cd stuff/state/
lftp chris@localhost:~/stuff/state> ls
-rw-r--r--    1 13597    13597        1394 Oct 23  2014
enrolled-20141012
-rw-r--r--    1 13597    13597         514 Oct 23  2014
enrolled-20141013
lftp chris@localhost:~/stuff/state> !pwd
/root
lftp chris@localhost:~/stuff/state> get survey-20141023.txt
3108 bytes transferred
lftp chris@localhost:~/stuff/state> put /etc/hosts
201 bytes transferred
lftp chris@localhost:~/stuff/state> ls
-rw-r--r--    1 13597    13597        1394 Oct 23  2014
enrolled-20141012
-rw-r--r--    1 13597    13597         514 Oct 23  2014
enrolled-20141013
-rw-r--r--    1 0        0             201 May 03 20:22 hosts
lftp chris@localhost:~/stuff/state> !ls
anaconda-ks.cfg          bin           install.log
dog                      Pictures      sent
Downloads                Public        survey-20141023.txt
lftp chris@localhost:~/stuff/state> quit
```

After providing the username (-u chris), lftp prompts for chris's Linux user password. Typing pwd shows that chris is logged in to the local host and that /home/chris is the current directory. Just as you would from a regular Linux command-line shell, you can use cd to change to another directory and ls to list that directory's contents.

To have the commands you run interpreted by the client system, you can simply put an exclamation mark (!) in front of a command. For example, running !pwd shows that the current directory on the system that initiated the lftp is /root. This is good to know because if you get a file from the server without specifying its destination, it goes to the client's current directory (in this case, /root). Other commands you might run so that they are interpreted by the client system include !cd (to change directories) and !ls (to list files).

Assuming that you have read permission for a file on the server and write permission from the current directory on the initiating system, you can use the `get` command to download a file from the server (`get survey-20141023.txt`). If you have write and upload permission on the current directory on the server, you can use `put` to copy a file to the server (`put /etc/hosts`).

Running an `ls` command shows that the `/etc/hosts` file was uploaded to the server. Running the `!ls` command lets you see that the `survey-20141023.txt` file was downloaded from the server to the initiating system.

## Using the gFTP client

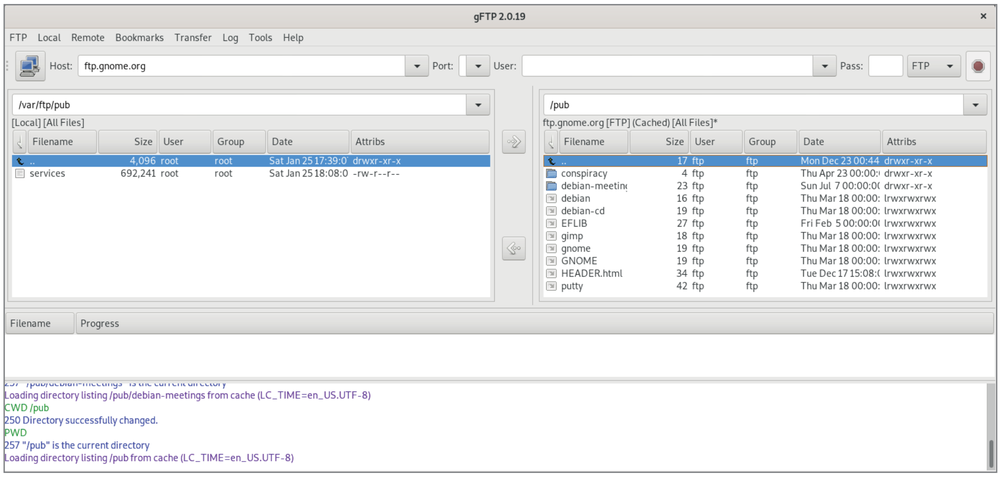
Many other FTP clients are available with Linux as well. Another FTP client that you could try is gFTP. The gFTP client provides an interface that lets you see both the local and remote sides of your FTP session. To install gFTP in Fedora, run the following command to install the `gftp` package:

```
# yum install gftp
```

To start gFTP, launch it from the applications menu or run `gftp &` from the shell. To use it, type the URL of the FTP server to which you wish to connect, enter the username you want to use (such as anonymous), and press Enter. Figure 18.3 shows an example of gFTP being used to connect to the `gnome.org` site: `ftp.gnome.org`.

**FIGURE 18.3**

The gFTP FTP client lets you see both sides of an FTP session.



Host: ftp.gnome.org

/var/ftp/pub
[Local] [All Files]

```
Filename        Size  User   Group  Date                Attribs
..             4,096  root   root   Sat Jan 25 17:39:0  drwxr-xr-x
services     692,241  root   root   Sat Jan 25 18:08:0  -rw-r--r--
```

/pub
ftp.gnome.org [FTP] (Cached) [All Files]*

```
Filename         Size  User  Group  Date                Attribs
..                 17  ftp   ftp    Mon Dec 23 00:44    drwxr-xr-x
conspiracy          4  ftp   ftp    Thu Apr 23 00:00:   drwxr-xr-x
debian-meetin      23  ftp   ftp    Sun Jul  7 00:00:0  drwxr-xr-x
debian             16  ftp   ftp    Thu Mar 18 00:00:   lrwxrwxrwx
debian-cd          19  ftp   ftp    Thu Mar 18 00:00:   lrwxrwxrwx
EFLIB              27  ftp   ftp    Fri Feb  5 00:00:0  lrwxrwxrwx
gimp               18  ftp   ftp    Thu Mar 18 00:00:   lrwxrwxrwx
gnome              19  ftp   ftp    Thu Mar 18 00:00:   lrwxrwxrwx
GNOME              19  ftp   ftp    Thu Mar 18 00:00:   lrwxrwxrwx
HEADER.html        34  ftp   ftp    Tue Dec 17 15:08:   lrwxrwxrwx
putty              42  ftp   ftp    Thu Mar 18 00:00:   lrwxrwxrwx
```

```
257 "/pub/debian-meetings" is the current directory
Loading directory listing /pub/debian-meetings from cache (LC_TIME=en_US.UTF-8)
CWD /pub
250 Directory successfully changed.
PWD
257 "/pub" is the current directory
Loading directory listing /pub from cache (LC_TIME=en_US.UTF-8)
```

To traverse the FTP site from gFTP, just double-click folders (just as you would from a file manager window). The full paths to the local directory (on the left) and remote directory (on the right) are shown above the listings of files and folders below.

To transfer a file from the remote side to the local side, select the file that you want from the right and click the arrow in the middle of the screen pointing to the left. Watch the progress of the file transfer from messages on the bottom of the screen. When the transfer completes, the file appears in the left pane.

You can bookmark the address information that you need to connect to an FTP site. That address is added to a set of bookmarks already stored under the Bookmarks menu. You can select sites from the list to try out the gFTP. Most of the sites are for Linux distributions and other open source software sites.

## Summary

Setting up an FTP server is an easy way to share files over a TCP network. The Very Secure FTP Daemon (`vsftpd` package) is available for Fedora, Red Hat Enterprise Linux, Ubuntu, and other Linux systems.

A default `vsftpd` server allows anonymous users to download files from the server and regular Linux users to upload or download files (provided the correct security settings are applied). Moving around on an FTP server is similar to moving around a Linux filesystem. You move up and down the directory structure to find the content that you want.

There are both graphical and text-based FTP clients. A popular text-based client for Linux is `lftp`. As for graphical FTP clients, you can use a regular web browser, such as Firefox, or dedicated FTP clients, such as gFTP.

FTP servers are not the only way to share files over a network from Linux. The Samba service provides a way to share files over a network so that the shared Linux directory looks like a shared directory from a Windows system. Chapter 19, "Configuring a Windows File Sharing (Samba) Server," describes how to use Samba to offer Windows-style file sharing.

## Exercises

The exercises in this section describe tasks related to setting up an FTP server in RHEL or Fedora and connecting to that server using an FTP client. If you are stuck, solutions to the tasks are shown in Appendix B. Keep in mind that the solutions shown in Appendix B are usually just one of multiple ways to complete a task.

Don't do these exercises on a Linux system running a public FTP server because they almost certainly will interfere with that server.

1. Determine which package provides the Very Secure FTP Daemon service.
2. Install the Very Secure FTP Daemon package on your system, and search for the configuration files in that package.

3. Enable anonymous FTP and disable local user login for the Very Secure FTP Daemon service.
4. Start the Very Secure FTP Daemon service and set it to start when the system boots.
5. On the system running your FTP server, create a file named `test` in the anonymous FTP directory that contains the words "Welcome to your vsftpd server."
6. From a web browser on the system running your FTP server, open the `test` file from the anonymous FTP home directory. Be sure that you can see that file's contents.
7. From a web browser outside of the system that is running the FTP server, try to access the `test` file in the anonymous FTP home directory. If you cannot access the file, check that your firewall, SELinux, and TCP wrappers are configured to allow access to that file.
8. Configure your `vsftpd` server to allow file uploads by anonymous users to a directory named `in`.
9. Install the `lftp` FTP client (if you don't have a second Linux system, install `lftp` on the same host running the FTP server). If you cannot upload files to the `in` directory, check that your firewall, SELinux, and TCP wrappers are configured to allow access to that file.
10. Using any FTP client you choose, visit the `/pub/debian-meetings` directory on the `ftp.gnome.org` site and list the contents of that directory.